5-5-2016

The Consensus Problem, Cellular Automata, and Self-Replicating Structures

David Griffin
Utah State University

The Consensus Problem, Cellular Automata, and Self-replicating Structures.

David Griffin and Dr. David Peak

Abstract:

Over The course of the last four years I have researched the consensus problem. I have done so by studying how cellular automata following the 2DGKL rule are able to reach consensus in a verity of ways. There are only certain structures that can form within a network, and these structures can be described and examined directly from the rules that make them up. I have also explored a variety of methods to study the rule including, graph theory and liner algebra representations of the cellular automata. Additionally I collected an analyzed data on symmetry in the rule, dependency on size and boundary conditions. I also have studied alternative rules.

Background:

A cellular automata network is a computing paradigm that consists of a series of nodes. Each node or automaton has some internal state. The cellular automata I used could only be in the states 'on' or 'off'. Each node also has some number of neighboring nodes that are able to communicate their internal state. The automata then are able to update according to some rule based off their own state and the states of their neighboring nodes.

There is no central processing unit connecting the cellular automata. Each node simply follows a rule that tells it how to update according to its neighboring nodes' state and its own. When there is a network of these automata all following a rule, the whole network is able to perform a global task. This is called emergent computation. I am studying cellular automata that are often able to perform the majority task. The network is given an initial configuration that specifies what state each node is in. If the network is able to successfully perform is the majority task each node will eventually update to the state initially in the majority.

The cellular automata will be arranged in a square lattice. The node of interest will be refer to as the Central(C) node. Each node has four neighbors that for ease we will refer to as East (E), North (N), West (W), and South (S). We will also on refer to the nodes that are diagonal from a central node as Northeast (NE), Northwest (NW), Southeast (SE) and Southwest (SW). I studied networks with finite size which wrap around on the boundaries such that the topological is a torus. I later explored networks with other boundary conditions.

I mostly studied the 2DGKL rule or the directional majority rule. In the directional majority rule each node updates to the majority state within a focus area that is determined by the internal state of the node. If the central node is 'on' then the focused area is C, N, and W. The focuses area for a node that is

'off' is C, S, and E. There are four rules that are symmetrically the same which can be created by swapping N with S or E with W.

Notice if a node is 'on' then for it to update to 'off' both N and W must be 'off'. The same can be said for 'off' with South and East as 'on'. It only takes one neighbor in a node's focus area to be in the same state as the node to keep a node from changing its state.

Digraph representation:

While studying these cellular automata I considered each automata to be a node of a digraph with edges showing how information flows for a specific state. It is convenient to consider one states digraph representation at a time. For example in the on-digraph representation of a network, every node in the 'on' state would have edges leading to them from nodes that are N and W. The digraph for a state is defined such that if a node is in some state and in the digraph representation for that state there is an edge leading to that node from a neighbor in the same state then the node will update to the same state; otherwise, it will change states.

The digraph representation has its limitations. The network changes every time step, and I was unable to fine an equivalent graph representation that was time dependent.

Stable Networks and Regions:

To study these networks I looked to structures that formed within the network as precursors to the final state. A network can either be stable or unstable. A stable network is one in which the network either doesn't change or leads to a configuration which will eventually lead back to the original. An unstable network will never return back to its current configuration. Finite size network must be a finite number of configurations. Each configuration leads to the next in a deterministic way; thus, any configuration will eventually lead to a stable network. Each node in a stable configuration can be said to be steady (non-changing) or periodic. A sub area or region of a network can also be said to be stable if the nodes within the region all are steady or periodic and can be shown to be stable independent of what the nodes outside the region do. This is known as self-replicating structure because the structure recreates itself each time step.

Bounded Regions:

A region is bounded if it does not contain the whole network. Boundary nodes are nodes that have neighbors outside of the region. The state of these nodes does not have any effect on the state of the nodes within a steady region.

Steady cycles:

The first self-replicating structure I found is a steady cycle. A node must be steady if it is in some state, and one of the nodes in its focus area is steady in the same state. If there is a cycle in the graph representation of the network for one of the states, then each node in the cycle must be steady. This is true even if the network is not yet stable. These cycles usually form well before the rest of the network reaches stability. These cycles are an example of a steady region. These steady regions can also include

any node that is the same state as the cycle that is also reachable from the cycle in the graph representation of the network for that state states as long as the region also includes all the nodes along a path from the cycle to that node.

Bounded periodic cycles:

A bounded region is called periodic if any of the nodes in the region are periodic. I found two types of periodic regions; linear periodic cycles and non-linear periodic cycles. Both will be discussed in the following section as well as proof that they are the only periodic regions. This will be done by assuming a region is periodic then exploring what can be known about the region based on only that.

Lemma 1:

If a periodic node is part of a stable region then it must also contain all of its neighbors.

A periodic node C at some point must change from 'on' to 'off', thus N and W must be 'off' at that time. Also the periodic node must change from 'off' to W, thus S and E must be both 'on' at some point. This shows that for a stable region of a network to contain a periodic node it must also contain all of that nodes neighbors.

Lemma 2:

Any periodic node in a stable network or stable region of the network has at least one of N, E, or NE, hat is also periodic, as well as at least one of S, W, or SW.

If a node is periodic then E must in the 'on' state every time the periodic node is about to change from 'off' to 'on'. Thus E must be periodic or steady 'on' for the network to be stable. In the case that E is steady 'on' then NE must be periodic or steady 'on'. If NE is ever off at the same time as C then E will change from 'on' to 'off'. Thus NE or periodic or Steady 'on'. Using the same logic, N must periodic or steady 'off', and further more NE must be periodic or steady 'off', if N is steady 'off'. Finally if C is periodic, and E and N are both steady, then NE must be both periodic or Steady 'on' and periodic or Steady 'of'. The only solution is that NE is periodic.

Thus if C is periodic then N, E or NE must periodic. The same steps can be used to show that at least one of S, W, or SW must also be periodic.

Theorem 1:

Any periodic region will span the network. This is called a cycle.

From lemma 2 for a region to contain a periodic node it must contain a node to the north, east, or northeast. Because there are a finite number of nodes this must eventually wrap around back to the original.

Lemma 3:

Any stable region that is bounded, must only have steady nodes on the boundaries.

A stable region that contains a periotic node must contain all of its neighbors, lemma 1. For a node to be on the boundary it must be stable.

Lemma 4:

A bounded stable region that contains periodic nodes, must have a periodic node with N or E steady 'off'.

Assume a bounded stable region that contains periodic nodes but no periodic node such that N or W is steady 'off' then the whole network is periodic this is a contradiction as we assumed a bounded stable region. Thus there must be periodic nodes with N or W steady 'off'.

Corollary to Lemma 4:

In a bounded stable region that contains periodic nodes, must have a periodic node with S or E steady 'on'.

From Symmetry and lemma 3 this is obvious.

Lemma 5:

The northern end of a western border of a periodic region may not have a corner turning to the west, but may turn east.

Assume C is a periodic node in a stable region with W steady 'off'. Thus C is part of a periodic region and W is part of a border. N must be periodic or stable 'off'.

If N is periodic, then NW must be steady 'off' or periodic. Because W is steady 'off', NW can never change from 'off' to 'on' and must be steady off. And the boundary continues to be to the east as we move upwards.

In the Case that N is steady 'off' the boundary becomes a northern boundary, and the corner turns east. This shows that a western border cannot have a corner on the northern end that turns west. The border must turn east and become a northern border if it has a corner.

Corollary to Lemma 5:

The southern end of an Eastern border of a periodic region may not have a corner turning to the east but not west.

Lemma 6:

The eastern end of a northern border of a periodic region must have a corner to the north.

Assume C is a periodic node in a stable region with N steady 'off'. Thus C is part of a periodic region and N is part of a border. E must be periodic or stable 'on'. In either case, for N to remain 'off' NE must be periodic or 'off'. For E to be stable 'on' NE must be periodic such that it is never 'off' or 'on' at the same

time as C. Thus the Eastern end of a Northern border must be a corner to the north forming a new western border.

Corollary to Lemma 6:

The western end of a southern border of a periodic region must have a corner to the south.

Lemma 7:

Periodic nodes along a linear, cyclical boundary have a period equal to some factor of the length of the boundary.

Assume C is periodic with nodes to the North steady 'off' in a cyclical boundary.

Thus E is periodic and NE be steady 'off'. Every time C changes from 'off' to 'on' E must be 'on' furthermore E must change from 'on' to 'off' because C is 'off'.

Because the boundaries are cyclical this pattern will continue until it reaches its self. This means that every time a node changes from 'on' to 'off' then it will repeat in at most the length of the of the boulder. Then the period of the nodes along the boundary are some factor of the length of the boundary.

The same argument can be symmetrically applied to boundaries to each direction.

Lemma 8:

Periodic nodes along non-linear boundaries are only in the state opposite to the boundary for one time step.

If C is periodic with N and E steady 'off' in a cyclical boundary, then C can only be 'on' for one time step.

If C is periodic with N steady 'off' and E Periodic such that it is only ever 'on' for one time step. Then C must be 'on' the time step before E is 'on' and change to 'off' afterwards. Furthermore if C becomes 'on' when E is 'off' then it must change back to 'off' immediately.

This show that node along the north boundary in a periodic region are only 'on' for one time step. The same thing can be shown for western boundaries. Additionally symmetric arguments can be made that periodic nodes along South and East boundaries can only be 'off' for one time step.

Lemma 9:

A periodic node that is 'off' for more than time step in a row must have a node to the south or east that also periodic and 'off' for more than one step.

For C to change from 'off' to 'on' both S and E must be on at the step. At least one of S and E must be 'off' for C to remain 'off' for more than one time step. If that node were 'on' before C changed to 'off' then it would remain 'on'. This shows that that S and E must be 'off' for more than one step if C is periodic and 'off' for more than one time step.

This can be extended to show that if a periodic node that is ‘on’ more than one time step then either N or W must also be periodic that is ‘on’ more than one time step.

Theorem 2:

Bounded stable region that contain periodic nodes are formed with are not must have rectangle periodic regions with diagonal northeast and southwest corners.

Let C be a periodic N Steady ‘off’ and NE periodic. The region is clearly not linear. From lemma 9 and 8 C and NE must change state every time step, they must also always be opposite one another. E must be steady ‘on’. This means that E is an eastern boundary to C and a southern boundary NE. Thus northern boundary to meet eastern boundaries with another periodic region to the north east. Likewise western boundaries meet southern boundaries with another periodic region to the south west. Lemma 5 shows that southern boundaries meet eastern boundaries and northern meet western boundaries with no restriction on there being boundaries to the northwest or southeast.

Thus bounded periodic regions must be rectangles and must connect diagonally with another periodic regions. Additionally from lemma 8 and 9 it is clear that these bounded periodic nodes must all change state every time step. These stable periodic region must form cycles because of the finite size the requirement for them to connect diagonally to another one.

Cycles and stable regions:

I have found three types of stable regions: Steady cycle, linear periodic cycle and non-linear periodic cycle.

Steady cycles are formed from entirely steady nodes. They move from southeast to northwest. They can be linear and only move from south to north or east to west, but they are fundamentally the same as the non-linear type.

Linear periodic cycle are bounded to the north and south or east and west by linear steady cycles. Periodic nodes have a period that is some factor of the length of the border.

It has been demonstrated that any steady periodic node must be part of a periodic cycle. It can be shown that for a steady node to be part of a stable region of the network it must be part of a steady cycle, part of the boundary of periodic cycle, or have another stable node in its focus area.

The formation of these cycles allow for there to be stable areas in the network which grow as nodes just outside the stable areas become reachable from the cycles. In randomly constructed networks these cycles often form much quicker than it takes for the rest of the network to reach stability. The formation of these cycles is an integral part of the way in which 2DGKL rule actual works and reaches stability.

Linear algebra representation:

It is posable to make a large truth table that maps every posable network configuration as input to every posable configuration after one time step as output. Any truth table can be changed into a sum of products of a product of sums. The Sum of products for each node is actually quite small. I hoped I could find a way to represent the rule in a matrix representation which a computer could take to some power and find what happens after so many steps. The advantage of approaching the issue in this way is that it would allow the computer to calculate what N time steps in the future for any initial configuration. This might also allow me to find a singular value decomposition a limit that might in turn show something fundamental about the problem. I found the following representation but has its limitations.

Instead of one matrix to act as the operator, two are used. These matrix are fundamentally different, but both operate on an array of 1's and 0's and both return an array of 1's and 0's.

Sum Matrix:

Works like a normal matrix with the following rules:

0+0=0, 0+1=1, 1+0=1, 1+1=1

0*0=0, 0*1=0, 1*0=0, 1*1=1

Product Matrix:

Works like a normal matrix with the following rules:

0+0=0, 0+1=0, 1+0=0, 1+1=1

0*0=1, 0*1=1, 1*0=0, 1*1=1

Because they act fundamentally different they cannot be simplified into one matrix except in rare cases. A sum matrix operating on a sum matrix simplifies to a single sum matrix. This is done in a straight forward way. A product matrix operating on a product matrix simplifies to a single product matrix but this is done slightly differently than the way it operates on an input vector. A product matrix acts on a product matrix in the same way that a sum matrix acts on a sum matrix or an input vector.

To be able to make any set of outputs for a set of given inputs using product of sums or sum of products, both the inputs and their inverse are needed. Because we want to plug the output of one pair of matrix operations into another, the out puts should be the same.

The update using the 2DGKL for a single node(C) the sum of products look like this:

$$C' = (CN) + (CW) + (\bar{C}SE)$$

$$\overline{C'} = (\overline{CS}) + (\overline{CE}) + (C\overline{NW})$$

For a network of size n the product matrix will be 6n by 2n and the sum matrix will be 2n by 6n.

If $I_0$is the initial configuration and P is the right product matrix and S is the right sum matrix.

$SPI_0 = I_1$ and $(SP)^n I_0 = I_n$ where I_n is the configuration after n time steps. There must be some S_n and P_n such that $S_n P_n I_0 = I_n$.

$S_n P_n = (SP)^n$

A sum of products can be changes into a product of sums using De Morgan's law. Let $S'P'$ have the equivalent action on a set of inputs as SP where S' is a product matrix and P' is a sum matrix. Applying De Morgan's law is straight forward, because the inputs into P will have both the state of a node and its inverse and the results of S will have both the next state of a node and its inverse. To change from SP to $S'P'$ the columns of P corresponding to each nodes state must be swapped with those that correspond to that node's inverse, and the resulting P' matrix must be treated as a sum matrix. The rows of S corresponding to the next state must likewise be swapped with the rows corresponding to their inverse.

Note that this change from SP to $S'P'$ only works because the inputs and outputs are standardized to contain both a value and its inverse. More general sum and product matrix cannot be changed in the same way. Furthermore, SP to $S'P'$ only have the same results for an input vector if the input is formatted in this way.

$(SP)^2 = SPSP = S'P'SP$

P' and S are both sum matrixes and can be simplified down to a single sum matrix $Q = P'S$.

$(SP)^3 = \mathrm{SP}SPSP = SPS'QP = SQ'QP$ where $Q' = PS'$.

If there were a way to change from $SQ'to\ S'Q$ then $(SP)^n = S'Q^{n-1}P$.

This would have many advantages but unfortunately converting $SQ'to\ S'Q$ is not so simple. S, and thus Q, do not have the same requirement on their inputs that they must come in value-inverse pairs the way P does. Applying De Morgan's law is not so simple. There may still be away to accomplish this, but it will require further study.

Symmetry:

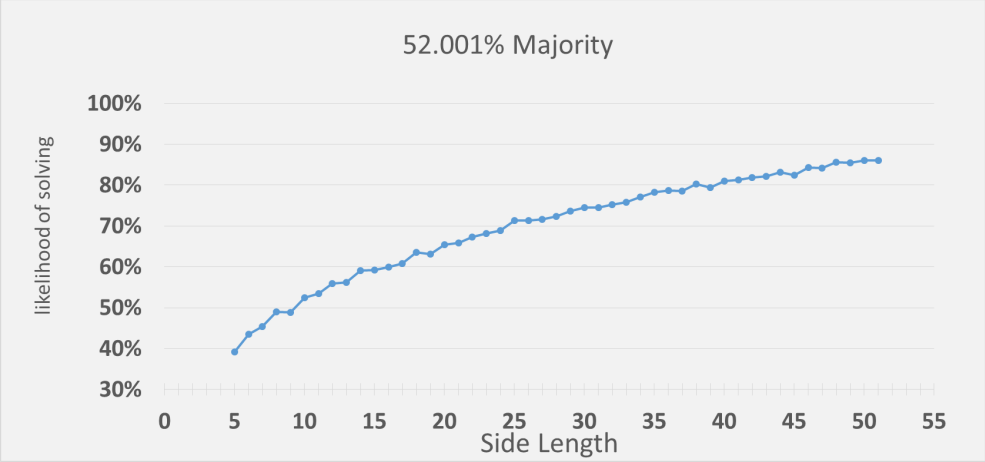
As noted above there is a strong symmetry in the 2GKL rule. In fact, there are four equivalent rules depending on what the focuses area of a state is; though, for a specific network each symmetric rule might have differing outcomes. I studied the relationship between these rules. Below is shown a pair correlation between the results in one 2DGKL rule and a symmetric one. The columns correspond to a known resolute of the network in one symmetry the rows correspond to the likelihood of a specific result in another symmetry. This data was collected from 10,000 initial configurations on a 15 by 15 network with 113 'on' nodes and 112 'off' nodes.

Pair correlation	Solves Right	Solves Wrong	Does Not Solve

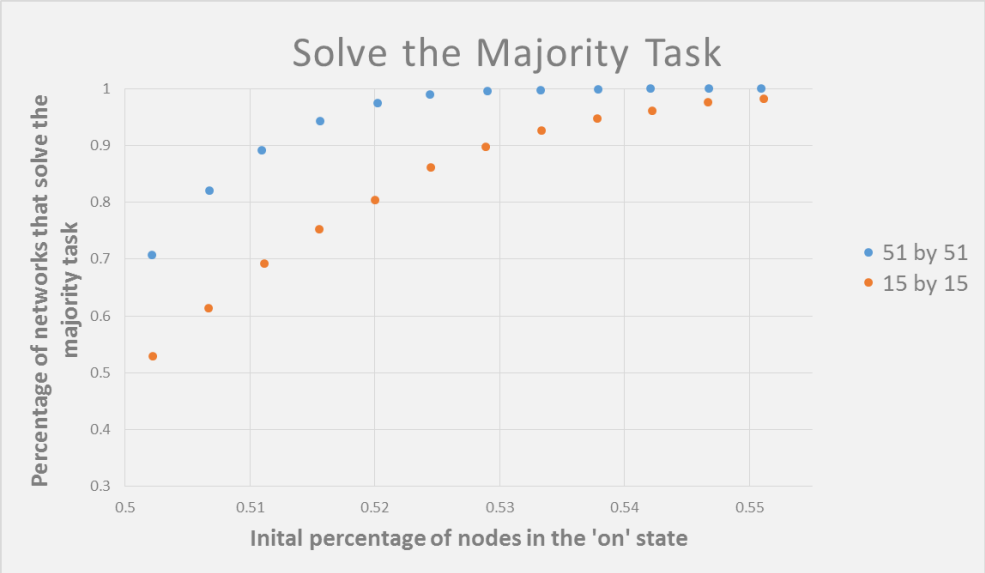
Solves Right	45.8%	25.4%	32.5%
Solves Wrong	21.3%	41.0%	27.7%
Does Not Solve	32.9%	33.6%	39.8%

Side length Dependency:

There is a strong dependency how often a network solves and the size of the network. To Study this relationship I ran tens of thousands of experiments with different side lengths. The graph below shows the percentage of networks that solve as a function of the side length.

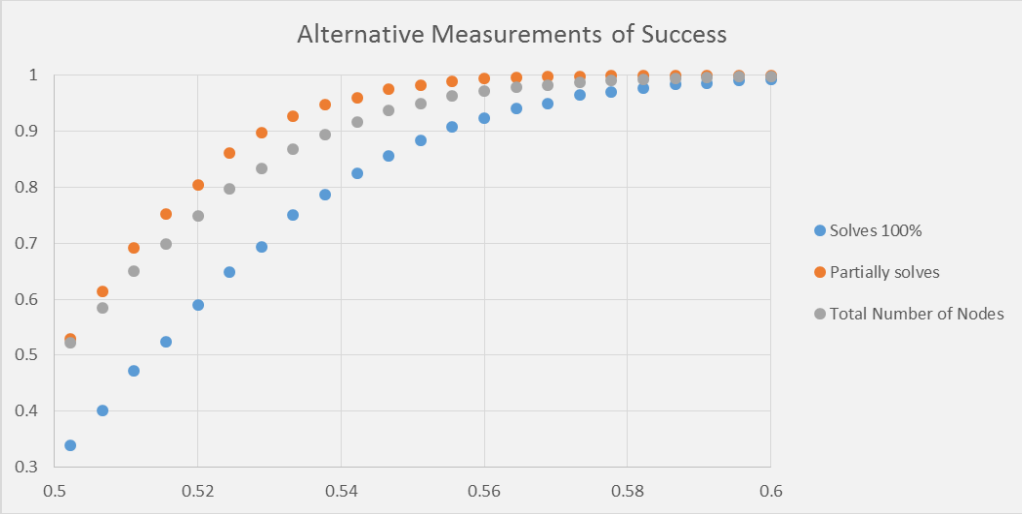


This graph shows how often a network solves as a function of the initial percentage in the majority. For both 51 by 51 networks and 15 by 15 networks. 10000 initial configurations were used in both cases.



Alternative Measurements of Success:

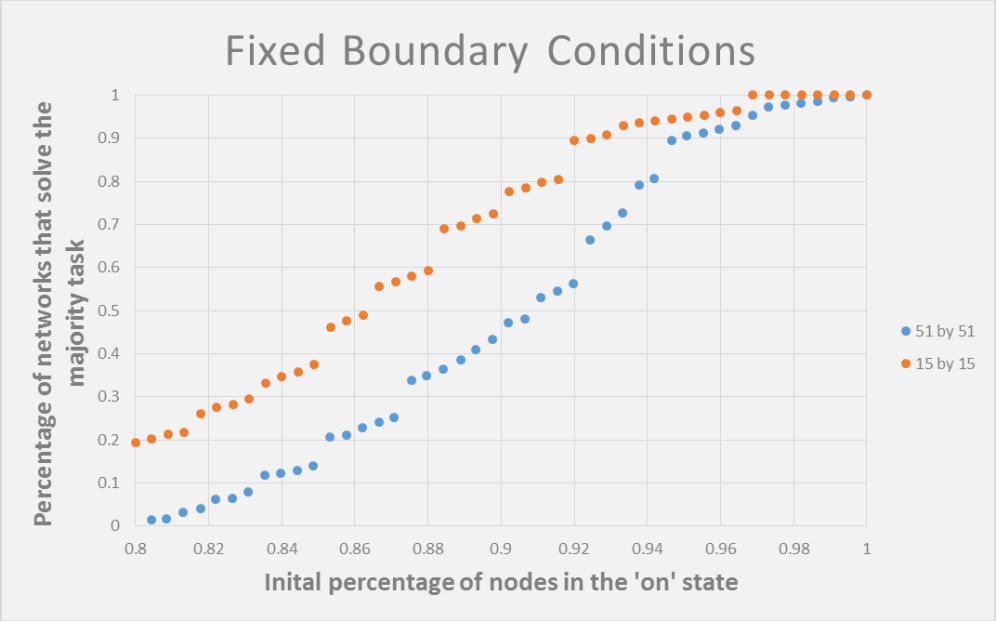
There are other ways to measure the success of a rule. One may consider networks that partially solve meaning that once they reach a stable condition they have more nodes in the majority state. One may also simply consider the total number of nodes. I examined the relationship between these measurements of success. Below is shown a graph comparing each method for a 15 by 15 network.



Fixed Boundaries:

The cyclic boundaries are not the only boundaries conditions that are worth looking at. Another boundaries condition, that is often used, is to let there be nodes just outside the boundaries which are fixed and never update. Each point of data represents the average of 10000 networks ran with 4 different fixed boundaries. Notice that these need a majority that has at least 80% of the nodes to start

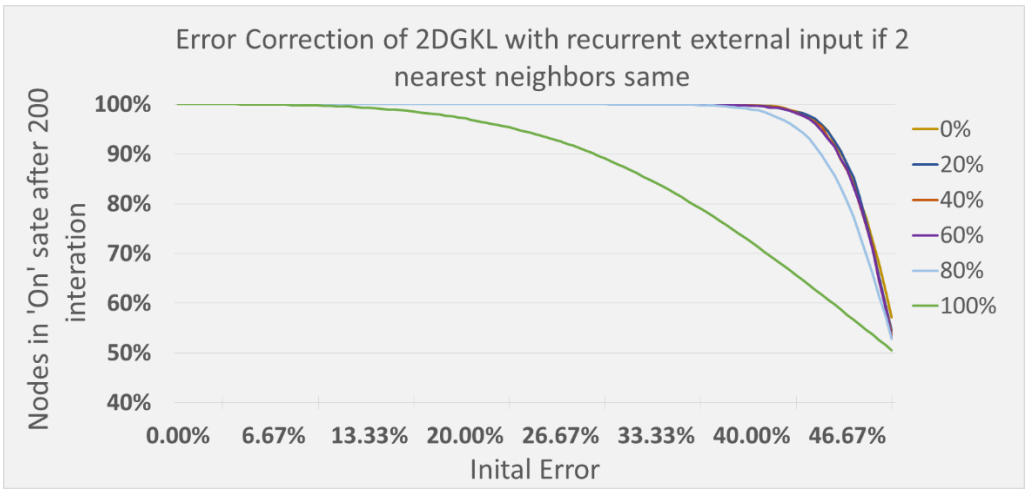
to solve whereas the cyclic boundary condition solve almost 100% of the time when there was only 60% of the nodes in the majority.

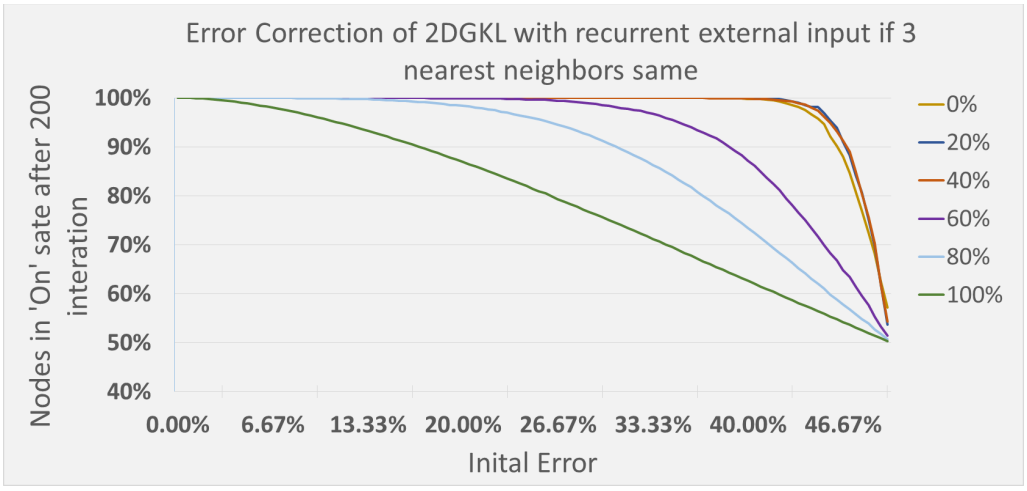


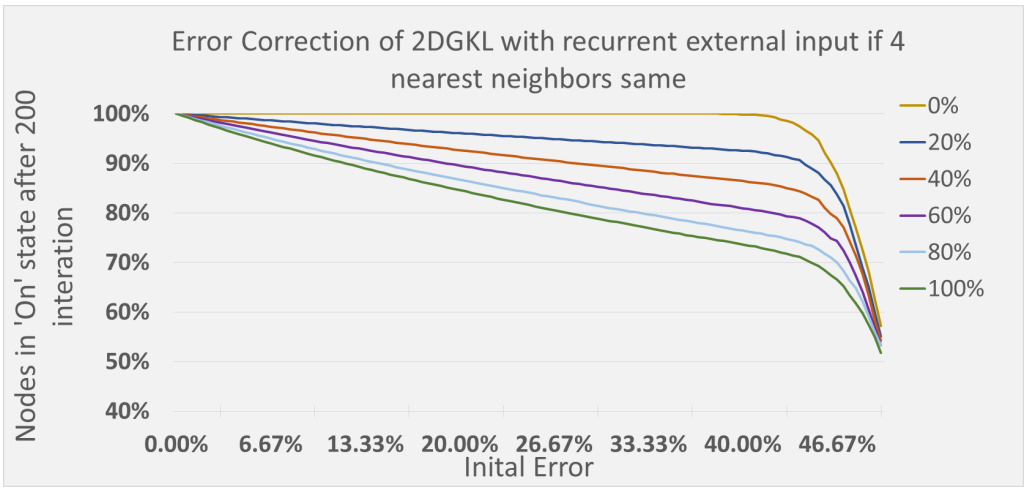
Error Correction:

One can look at these networks under the paradigm that they have received input from the outside world as their initial configuration and that they then try to reach a consensus of what that input was. An interesting question to ask is what happens if a node at random changes back to its initial state. In this experiment each so that in certain situations a node would have some probability of reverting to its original state instead of following the rule. The situations were based on how many of the neighboring nodes were in agreement independent of what state the central node was in.

On each of the graphs below, the average number of nodes after 200 iterations is shown as a function of the initial error, or percentage of nodes in the minority. Each line represents the probability of a node reverting to its initial state in the situation being tested.


Error Correction of 2DGKL with recurrent external input if 2 nearest neighbors same
Nodes in 'On' sate after 200 interation
100%
90%
80%
70%
60%
50%
40%
0.00%
6.67%
13.33%
20.00%
26.67%
33.33%
40.00%
46.67%
Inital Error
0%
20%
40%
60%
80%
100%


Error Correction of 2DGKL with recurrent external input if 3 nearest neighbors same
Nodes in 'On' sate after 200 interation
100%
90%
80%
70%
60%
50%
40%
0.00%
6.67%
13.33%
20.00%
26.67%
33.33%
40.00%
46.67%
Inital Error
0%
20%
40%
60%
80%
100%


Error Correction of 2DGKL with recurrent external input if 4 nearest neighbors same
Nodes in 'On' state after 200 interation
100%
90%
80%
70%
60%
50%
40%
0.00%
6.67%
13.33%
20.00%
26.67%
33.33%
40.00%
46.67%
Inital Error
0%
20%
40%
60%
80%
100%

Other Paradigms:

I attempted to find other paradigms to use on this problem. This included trying to add hidden variables to allow the network to be reversible, considering it as a relaxation problem, and comparing it to the ising problem. I was not able to find another representation that added more insight in to the issue, but I believe there must be representation which would give great insight into how the cellular automata is able to perform its task. Valuable future research should be done to searching for such a representation.

Other Rules:

One of the simplest rules to perform the majority task is the Local Majority Rule (LMR) in which each node simply updates to the state that is in the majority between itself and its neighbors. This rule hardly ever solves except in extreme situations. With a small modification this rule can be made to work. The Modified Local Majority Rule (MLMR) follows the Local Majority Rule except when both N and W are 'off' and E and S are 'on', central node changes state. MLMR has many similarities to 2DGKL including four symmetric rules. More research should be done to find out how the MLMR compares to 2DGKL and to figure out how it works.

References:

D. Griffin and D. Peak, Emergent error-correction in distributed controller networks, Bull. Am. Phys. Soc., 58, G1.00020 (2013).

D. Peak, J.D, West, S.M. Messinger, and K.A. Mott, Evidence for complex, collective dynamics and emergent, distributed computation in plants, Proc. Nat. Acad. Sci. USA, 101, 918-922 (2004).

Couzin, I. D., Ioannou, C. C., Demirel, G., Gross, T., Torney, C. J., Hartnett, A., ... & Leonard, N. E. (2011). Uninformed individuals promote democratic consensus in animal groups. science, 334(6062), 1578-1580.

Creutz, M. (1986). Deterministic ising dynamics. Annals of physics, 167(1), 62-72.

J.P. Crutchfield and M. Mitchell, The evolution of emergent computation, Proc. Nat. Acad. Sci. USA, 92, 10742-10746 (1995).

L. Chua and L. Yang, Cellular neural networks: Theory, IEEE Trans. on Circuits and Systems, 35, 1257-1272 (1988).

Miller, M. B., & Bassler, B. L. (2001). Quorum sensing in bacteria. Annual Reviews in Microbiology, 55(1), 165-199.

Sethna, J. P. (2006). Statistical mechanics: entropy, order parameters, and complexity. Oxford: Oxford University Press.

S.M. Messinger, K.A. Mott, and D. Peak, Task-performing dynamics in irregular, biomimetic networks, Complexity 12, 14-21, 2007.

S. Wolfram, Statistical mechanics of cellular automata, Rev. Mod. Phys., 55, 601-644 (1983).